1. What exactly is []?

**Ans.** The empty list value, which is a list value that contains no items. This is similar to how '' is the empty string value.

you can add elements to an empty list like this:

my\_list = []

my\_list.append(10)

my\_list.append("hello")

my\_list.append([1, 2, 3])

Now, my list contains three elements: the integer 10, the string "hello", and the list [1, 2, 3].

2. In a list of values stored in a variable called spam, how would you assign the value 'hello' as the third value? (Assume [2, 4, 6, 8, 10] are in spam.)

Let's pretend the spam includes the list ['a', 'b', 'c', 'd'] for the next three queries.

**Ans.** spam[2] = 'hello' (Notice that the third value in a list is at index 2 because the first index is 0.)

spam = [2, 4, 6, 8, 10]

spam[2] = 'hello'

print(spam) # Output: [2, 4, 'hello', 8, 10]

1. How would you assign the value 'hello' as the third value?

spam = ['a', 'b', 'c', 'd']

spam[2] = 'hello'

print(spam) # Output: ['a', 'b', 'hello', 'd']

1. How would you assign the value 'world' as the first value?

spam = ['a', 'b', 'c', 'd']

spam[0] = 'world'

print(spam) # Output: ['world', 'b', 'c', 'd']

1. How would you assign the value ['x', 'y', 'z'] as the fourth value?

spam = ['a', 'b', 'c', 'd']

spam[3] = ['x', 'y', 'z']

print(spam) # Output: ['a', 'b', 'c', ['x', 'y', 'z']]

3. What is the value of spam[int(int('3' \* 2) / 11)]?

**Ans.** 'd' (Note that '3' \* 2 is the string '33', which is passed to int() before being divided by 11. This eventually evaluates to 3. Expressions can be used wherever values are used.)

int('3' \* 2) evaluates to int('33'), which is converting the string '33' to an integer, resulting in the value 33.

Then, int('33') / 11 performs the division, resulting in 3.

Finally, spam[int(3)] becomes spam[3], which means we are accessing the element at index 3 in the list spam.

If the spam list is defined as ['a', 'b', 'c', 'd'], then spam[3] will give us the value 'd'.

So, the value of spam[int(int('3' \* 2) / 11)] is 'd'.

4. What is the value of spam[-1]?

**Ans**. Negative indexes count from the end

spam = [10, 20, 30, 40, 50]

print(spam[-1])

The output will be 50, which is the last element of the list spam. If spam is a string, it will work in a similar way, returning the last character of the string.

5. What is the value of spam[:2]?

Let's pretend bacon has the list [3.14, 'cat,' 11, 'cat,' True] for the next three questions.

**Ans.** The expression bacon [:2] is a slicing operation that extracts a sublist from the beginning up to, but not including, the element at index 2.

For the list bacon = [3.14, 'cat', 11, 'cat', True], bacon[:2] will give us the elements at indices 0 and 1, which are 3.14 and 'cat', respectively.

Let us look like this :

So, the value of bacon [:2] is [3.14, 'cat']. It returns a new list containing the first two elements of the original list bacon.

bacon = [3.14, 'cat', 11, 'cat', True]

result = bacon [:2]

print(result)

**Output**

[3.14, 'cat']

bacon [:2] performs slicing on the list bacon and extracts the elements from index 0 to index 1 (up to, but not including index 2). The result is a new list [3.14, 'cat'], containing the first two elements of the original list bacon.

6. What is the value of bacon.index('cat')?

**Ans.** Let's use the provided bacon list [3.14, 'cat', 11, 'cat', True] to determine the index of the first occurrence of 'cat':

bacon = [3.14, 'cat', 11, 'cat', True]

index\_of\_cat = bacon.index('cat')

print(index\_of\_cat)

**Output** = 1

the value of bacon.index('cat') is 1, as the first occurrence of the element 'cat' is at index 1 in the list bacon.

7. How does bacon. Append (99) change the look of the list value in bacon?

**Ans**. If bacon was initially an empty list or if it didn't contain the value 99 before, the append(99) operation will add the element 99 to the end of the bacon list.

For example, suppose bacon was initially [1, 2, 3, 4, 5]. After calling bacon.append(99), the bacon list would look like this:

The append() method modifies the list in place and doesn't return anything. It simply adds the specified element (99 in this case) to the end of the list bacon.

bacon.append(99)

print(bacon)

[3.14, 'cat', 11, 'cat', True, 99]

8. How does bacon. Remove('cat') change the look of the list in bacon?

**Ans.** bacon.remove('cat')

print(bacon)

[3.14, 11, 'cat', True, 99]

9. What are the list concatenation and list replication operators?

**Ans.** The operator for list concatenation is +, while the operator for replication is \*. (This is the same as for strings.)

10. What is difference between the list methods append () and insert ()?

**Ans.** insert() can add them anywhere in the list, While append() will add values only to the end of a list.

11. What are the two methods for removing items from a list?

**Ans.** The del statement and the remove() list method are two ways to remove values from a list.

12. Describe how list values and string values are identical.

**Ans.** Both lists and strings can be passed to len(), have indexes and slices, be used in for loops, be concatenated or replicated, and be used with the in and not in operators.

13. What's the difference between tuples and lists?

**Ans**. Lists are mutable; they can have values added, removed, or changed. Tuples are immutable; they cannot be changed at all. Also, tuples are written using parentheses, ( and ), while lists use the square brackets, [ and ].

14. How do you type a tuple value that only contains the integer 42?

**Ans.** (42,) The trailing comma is mandatory.

15. How do you get a list value's tuple form? How do you get a tuple value's list form?

Ans. The tuple() and list() functions, respectively

16. Variables that "contain" list values are not necessarily lists themselves. Instead, what do they contain?

Ans. They contain references to list values

17. How do you distinguish between copy. Copy () and copy. deepcopy()?

Ans. The copy.copy() function will do a shallow copy of a list, while the copy.deepcopy() function will do a deep copy of a list. That is, only copy.deepcopy() will duplicate any lists inside the list.